5
CONCLUSIONS AND FUTURE WORK

5.1 Summary of technical contributions

5.2 Summary of empirical findings

5.3 Future Work

Furthermore, WASM-MUTATE can benefit from the enumerative synthesis techniques employed by CROW and MEWE. Specifically, WASM-MUTATE could incorporate the transformations generated by these tools as rewriting rules.

Moreover, the WebAssembly ecosystem is still in its infancy compared to more mature programming environments. A 2021 study by Hilbig et al. found only 8,000 unique WebAssembly binaries globally[**?**], a fraction of the 1.5 million and 1.7 million packages available in npm and PyPI, respectively. This limited dataset poses challenges for machine learning-based analysis tools, which require extensive data for effective training. The scarcity of WebAssembly programs also exacerbates the problem of software monoculture, increasing the risk of compromised WebAssembly programs being consumed[**?**]. This dissertation aims to mitigate these issues by introducing a comprehensive suite of tools designed to enhance WebAssembly security through Software Diversification and to improve testing rigor within the ecosystem.

Program Normalization WASM-MUTATE was previously employed successfully for the evasion of malware detection, as outlined in [**?**]. The proposed mitigation in the prior study involved code normalization as a means of reducing the spectrum of malware variants. Our current work provides insights into the potential effectiveness of this approach. Specifically, a practically costless process of pre-compiling Wasm binaries could be employed as a preparatory measure for malware classifiers. In other words, a Wasm binary can first be compiled with

---

[0]Comp. time 2023/10/06 10:48:02

wasmtime, effectively eliminating approx. 25% of malware variants according to our preservation statistics for wasmtime. This approach could substantially enhance the efficiency and precision of malware detection systems.